2013 IEEE International Conference on Cloud Computing Technology and Science

Compliance-preserving Cloud Storage Federation based on Data-driven Usage Control

Tobias Wüchner
Technische Universität München, Germany
tobias.wuechner@cs.tum.edu

Steffen Müller and Robin Fischer
Karlsruhe Institute of Technology, Germany
{*st.mueller, robin.fischer*}*@kit.edu*

***Abstract*—Cloud storage federation improves service availability and reduces vendor lock-in risks of single-provider cloud storage solutions. Federation therefore distributes and replicates data among different cloud storage providers. Missing controls on data location and distribution however introduce security and compliance issues. This paper proposes a novel approach of using data-driven usage control to preserve compliance constraints in cloud storage federation. Based on common compliance regulations and laws we provide a brief categorization of compliance problems into spatial, temporal, and qualitative requirements. In addition, we show how usage control policies can be employed to constrain federation according to these categories. To demonstrate the feasibility of our approach we evaluate security and performance of our prototypical implementation.**

***Keywords*-cloud storage; federation; usage control; compliance;**

I. Introduction

Steadily increasing data volumes and the rising dependency on data ubiquity is one of the main reasons for the advent of cloud storage services. Today a plethora of cloud storage services, like DropBox or Google Drive, allow users to store their data on remote servers. These services generally offer superior availability and reliability due to redundancy and data distribution at provider side. Relying on one particular storage provider, however, implies a strong dependency on its service qualities like availability or costs. In case of service availability issues, for example, the availability of customer data is compromised. Mostly, leaving customers little compensation options.

Cloud storage federation addresses such single-provider problems through brokerage: users employ a federation service to take care of the data storage and maintenance at different providers, rather than storing data at only one location. This service then distributes and replicates the data among different cloud storage providers for the sake of reduced vendor lock-in and increased data availability.
Problem: Applying cloud storage federation can induce issues with data security and compliance requirements. Especially the transparent data distribution and replication on the provider-side limits the user's direct control over data flows, leading to potential violations of compliance constraints. Personal data, for example, sometimes must not leave the jurisdiction of a specific country. While the distribution in such a case is reasonable in terms of availability, it clearly can violate privacy compliance regulations like e.g. the EU Data Protection Directive.
Solution: This paper presents an approach to embody data-driven usage control and provenance tracking into a federated cloud storage system to enforce compliance constraints in form of usage control policies.
Contribution: Existing approaches to tackle federation compliance problems demand modifications to the underlying storage services or assume a certain storage architecture. Our approach in contrast acts as a light-weight, transparent proxy between storage users and providers, which allows us to enforce compliance in an unintrusive way, independent of underlying cloud storage infrastructures.
Organization: We provide required backgrounds on compliance requirements, cloud storage federation and data-driven usage control in section II. The data flow model and architecture of our approach are presented in section III. In section IV, we evaluate security and performance of our approach. We relate to existing work in section V. The paper concludes with a discussion of benefits and limitations in section VI.

II. Background

A. Compliance Requirements for Data Storage Services

Compliance means to act in correspondence with relevant laws, regulations, standards, and specifications. In the context of cloud storage services this relates to constraints on the storage, usage, distribution, or deletion of data. Based on an analysis of relevant compliance-related laws and regulations, we identify three categories of compliance requirements for our compliance enforcement approach:
Spatial Requirements: Federated cloud storage systems distribute data to multiple locations and thus potentially cross different jurisdictions. In this context, the EU directive 95/46/EC, e.g. requires personal data to not be stored in countries with inadequate level of privacy protection.
Temporal Requirements: Laws and regulations may demand that certain types of data must be maintained for or deleted after a given period of time. SOX, for example, requires audit (related) information to be maintained for at least 7 years. PCI-DSS, e.g., requires deletion of cardholder data after exceeds defined retention periods.
Qualitative Requirements: Laws and regulations put constraints on how data is handled. For example, access to or transport of data is constrained to be only granted when using certain security technology. PCI-DSS, e.g., requires to use SSL-based mechanisms to protect insecure services.

978-0-7695-5095-4/13 $31.00 © 2013 IEEE
DOI 10.1109/CloudCom.2013.149

B. Federated Cloud Storage

Cloud federation unites services from different providers into a single resource pool to increase redundancy or combine complementary resources.

Through different routing and replication techniques, cloud federation can improve availability, or ease migration. The open-source federated cloud storage service *MetaStorage* [1] acts as basis for our approach.

MetaStorage consists of *Node*, *Distributor* and *Coordinator* components. A Node acts as a wrapper for a cloud storage service and provides a generic interface of the cloud storage service to a Distributor component. A Distributor unites a set of nodes and is responsible for the replication and retrieval of the data as well as the assertion of the availability of replica. A Coordinator component manages the configuration states of different Distributor instances. Additionally, it handles the communication between multiple MetaStorage instances. The Coordinator instances are deployed within so-called *MetaStorageHosts*, containing at least one Coordinator and Distributor pair. User interaction with MetaStorage is handled through a SOAP interface (WSHandler) of the MetaStorageHost.

C. Data-driven Usage Control

Data-driven usage control combines usage control with dynamic data flow tracking to ensure a holistic enforcement. The Obligation Specification Language (OSL) is a general-purpose language to formally express usage obligations and provisions as first-order linear temporal logic predicates on sequences of events that express what should and must not happen to data in the future [2].

OSL policies come in form of Event-Condition-Action (ECA) rules that contain logic predicates on sequences of events. The *Event* part specifies for which incoming events the policy should be triggered, the *Condition* part contains further logical propositions on these events, and the *AuthorizationAction* part specifies the result of the policy evaluation process. An incoming Event consists of a name (*actionName*), a list of parameters (*paramMatch*), and attributes that specify the location of the issuer of the event (Loc_{src}) and that of its intended target (Loc_{dst}).

The *condition* part specifies predicate or temporal logic constraints on event parameters. The used notation also supports the specification of *xPath* expressions and state-based predicates, that express containment constraints between container and data. We enriched the OSL with a location-based operator *isLocatedWithin*, which evaluates to true iff a given event location is within a defined radius around a trigger location (Loc_{cmp}).

If an event triggers a policy and matches all specified constraints, the *AuthorizationAction* part is evaluated. It specifies, whether the event should be allowed, inhibited, or modified prior to its actual execution.

Due to its general-purpose applicability, we see OSL as natural candidate for the specification of compliance constraints and thus as basis for our approach. For details on its syntax and semantics please refer to [2], [3].

III. Approach

A. Data Flow Model

We model a cloud storage federation service as state machine that represents the current distribution of data among different virtual locations. Specific actions that are issued on the federation service (PUT, GET, DELETE) influence the distribution of data and thus modify the state of this model. A cloud storage federation service is thus defined by $(Data, Loc_{virt}, Action, R, \Sigma, i)$, where Σ is the set of states with $\Sigma := store \times own$, and $i \in \Sigma$ is the initial state with $i := \emptyset \times \emptyset$. The *store* function describes the containment relation between data items and storage locations. The *own* function represents the possession relation between data and entities. The transition relation $R \subseteq \Sigma \times Action \rightarrow \Sigma$ represents the state changes issued by the execution of PUT, GET, or DELETE actions. Additionally we define a function update notation: for any mapping $m : S \rightarrow T$ and variable $x \in X \subseteq S$, we define $m[x \leftarrow expr]_{x \in X} = m'$ with $m' : S \rightarrow T$ so that $m'(y) = expr$ if $y \in X$ and $m'(y) = m(y)$ otherwise. The semantics of this model is defined over traces that map abstract points in time to sets of states. Listing 1 depicts the definitions of the used sets and relations.
The transition relation R is then defined as follows:

A **PUT** action results in a flow of a data item from a client machine (clt) to a storage provider server (srv). The first PUT action at t=0 determines the data possession:

$$\forall store \in [Data \rightarrow 2^{Loc_{virt}}], \forall own \in [Data \rightarrow Ent],$$
$$\forall d \in Data, \forall srv, clt \in Loc_{virt}, \forall e \in Ent, \forall t \in \mathcal{N} \setminus \{0\} :$$
$$((store, own), (PUT, d, clt, srv, e, t),$$
$$(store[srv \leftarrow store(srv) \cup d], own)) \in R$$

A **GET** action is inverse to a PUT action and results in a data flow from a storage server to a client:

$$\forall store \in [Data \rightarrow 2^{Loc_{virt}}], \forall own \in [Data \rightarrow Ent],$$
$$\forall d \in Data, \forall srv, clt \in Loc_{virt}, \forall e \in Ent, \forall t \in \mathcal{N} :$$
$$((store, own), (GET, d, clt, srv, e, t),$$
$$(store[clt \leftarrow store(clt) \cup d], own)) \in R$$

A **DELETE** action removes a data item from a specific storage server (= virtual location):

$$\forall store \in [Data \rightarrow 2^{Loc_{virt}}], \forall own \in [Data \rightarrow Ent],$$
$$\forall d \in Data, \forall srv, clt \in Loc_{virt}, \forall e \in Ent, \forall t \in \mathcal{N} :$$
$$((store, own), (DELETE, d, clt, srv, e, t),$$
$$(store[srv \leftarrow store(srv) \backslash d], own)) \in R$$

B. Architecture

Our approach bases on event interception: prior to any execution of data manipulation events like GET, PUT, DELETE, the system verifies if the event execution would

Listing 1: Data Flow Model - Sets and Relations

$$Data := Name \times Content \times 2^{Tag}$$
$$Loc_{virt} := Name \times 2^{IP}$$
$$Ent := Person \cup Organization$$
$$Person := Name \times 2^{Role} \times Organization$$
$$Organization := Name \times 2^{Ent}$$
$$Action := \{PUT, GET, DEL\} \times Data \times Loc_{virt} \times Loc_{virt} \times Person \times Timestamp$$
$$store : Loc_{virt} \rightarrow 2^{Data}$$
$$own : Data \rightarrow Ent$$

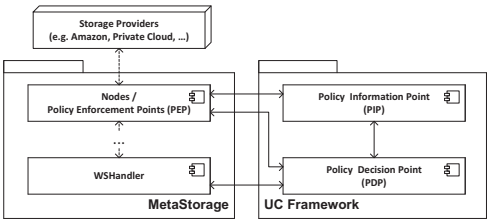

Figure 1: Conceptual Architecture

violate a deployed compliance policy. Based on the verification result it then either allows, inhibits, or modifies the event execution.

Events are directly intercepted at the level of *Node* components that wrap the target cloud storage providers and thus take care of the actual data distribution and deletion. Whenever a *Node* component tries to retrieve (GET), store (PUT), or delete (DELETE) data from a storage provider, the further execution is blocked and the event is forwarded to the employed usage control framework which then takes care of the actual decision process. The usage control framework, as described in detail in [2], is in charge of controlling the execution of usage-related events, based on previously deployed usage control policies. In our case, these policies represent compliance requirements. Therefore, prior to any compliance verification, a corresponding compliance policy must have been deployed to the usage control framework. Figure 1 depicts the high-level architecture of our integration approach.

The complete enforcement process looks as follows: i) the PEP intercepts all attempts to retrieve, store, or delete data from a cloud storage provider and forwards the corresponding events to the PDP; ii) based on previously deployed policies, the PDP then decides on their execution and modification, potentially querying the PIP, which implements the data flow model introduced in section III-A, for additional context information, and finally sends its decision back to the PEP; iii) the PEP then, based on the PDP's decision, either executes, modifies, or drops the event and, in case where the execution is not inhibited, notifies the PIP about the event execution; iv) the PIP then updates its internal state according to the modeled semantics of the intercepted event.

C. Instantiation

1) Compliance Policy Examples: Assume the following fictitious setting: ACME ltd. is a globally acting management consulting company. ACME recently moved parts of their data to the cloud to benefit from reduced data storage and maintenance costs. Fears concerning data availability and vendor lock-in issues drove the decision to employ a cloud storage federation solution. Next we depict some sample policies based on this scenario.

Example 1-Data Protection: ACME employees process large amounts of customer data, including personal data, to derive informed consulting advices. ACME thus underlies data protection regulations like the German data protection law. In consequence, data must not be stored on servers outside the EU. (Spatial requirement)

```
<preventiveMechanism name="Scenario1">
<trigger action="PUT"/>
<condition>
<not><isLocatedWithin>
  <loc_dst lon="50.555325" lat="10.085449" rad="2500"/>
</isLocatedWithin></not>
</condition>
<authorizationAction>
<inhibit/>
</authorizationAction>
</preventiveMechanism>
```

Example 2-Heterogeneous Protection Needs: ACME contractors have heterogeneous protection demands. ACME thus demands confidential data to be encrypted when stored in public clouds. (Quality requirement)

```
<preventiveMechanism name="Scenario2">
<trigger action="PUT"/>
<condition>
<not><impl>
 <xPathEval>
 contains(//event/param[@name="Data.Tag"]/@value,"conf")
 </xPathEval>
 <xPathEval>
 contains(//event/param[@name="Srv"]/@value,"private")
 </xPathEval>
</impl></not>
</condition>
<authorizationAction>
<modify>
  <parameter name="content" value="enc(AES,512,&cont)"/>
</modify>
</authorizationAction>
</preventiveMechanism>
```

2) System Implementation: We extended MetaStorage with functionality to enrich events with meta information like geographical location, or data ownership.

The geographical location of storage destinations and users is retrieved the Google geolocation API. Whenever an event execution attempt is detected, the geolocation API is queried to retrieve the location of the issuer and the target of the event. To identify to which entity a particular request is associated to, we use authentication information of MetaStorage, as requests to MetaStorage can only be issued by authenticated users. In general, all meta-information that is necessary to enforce the targeted compliance requirements could be either retrieved through the geolocation API, the MetaStorage API, or the different storage provider APIs.

IV. Evaluation

A. Security

Assumptions: Our major security assumption is that the employed usage control infrastructure can not be tampered or disabled. This assumption is crucial, as a client needs to trust the infrastructure to correctly enforce the deployed compliance policies, but it is justified as we assume our approach to be running at customer side.

Attacker model: An attacker in our setting is interested in interfering with the compliance enforcement mechanisms to issue non-compliant distribution or replication of data. As a direct manipulation of the enforcement infrastructure is ruled out by our assumptions, we only consider system-external attackers that only have access to the same

interfaces as regular non-malicious users.

Attacks and Countermeasures: One way an attacker could manipulate the compliance enforcement would be to deploy malicious policies into the system. This attack is counteracted by a built-in authentication and authorization system that restricts usage of the cloud storage federation service to authorized users. Another potential attack would be a man-in-the-middle-attack between client and federation service to e.g. on-the-fly change and weaken to-be deployed compliance policies. This attack is counteracted by the employed end-to-end encryption between all service components. Finally a denial-of-service could allow an authenticated attacker to flood the system with computationally complex policies to put stress on the evaluation component and slow down the service. One way of coping with this attack would be to introduce limits for the policy deployment frequency per user or data.

B. Performance

We tested our approach against a non-modified version of MetaStorage in terms of throughput and request latency. The client, the MetaStorage service, and the usage control framework were deployed on separate machines that were connected via a local network connection. This simulated a setting close to the anticipated application environment, where we assumed the components of our approach to be located within the same organizational boundaries.

To measure the performance we applied two test setups: First, we measured the performance of a non-modified MetaStorage service as base line; second, we evaluated the MetaStorage+ (with the compliance enforcement extension) with four deployed policies.

For every test setup we measured the minimum, the maximum, and the average latency in milliseconds. We set the test workload to 5,000 PUT requests, transferring about 3,300 different files, each of size 10kb to analyze the performance in a stress situation. For the second test setup with deployed policies, we tuned the requests to trigger a policy with a probability of 90%.

The results for MetaStorage and MetaStorage+ with deployed policies indicate an overhead of about 95% for the minimum (+98ms), 154% for the maximum (+791ms), 26% for the average latencies (+54ms), and a slight throughput penalty of about 20% (-0.26 kb/sec).

Being in general below the magnitude of 1, with an average latency penalty of less than 30%, we do not consider the performance overhead of our approach a show-stopper. In addition, we see potential for performance optimization of our implementation that could provide more satisfying performance in a realistic application context.

V. Related Work

The concept of CloudFilter by Papagiannis et al. [4] is close to ours as it intercepts HTTP requests to a cloud service to enforce file distribution policies. The main difference to this work is that we employ a sound usage control model to specify and enforce complex spatial, temporal, or qualitative compliance policies whereas CloudFilter bases its enforcement on pre-defined data labels. Furthermore, our approach explicitly tracks data provenance through a formal data flow model; CloudFilter does not.

Massonet et al. [5] propose a compliance enforcement architecture for a federated cloud infrastructure. The main difference to our work is that they rely on the RESERVOIR federation architecture with specific mechanisms in place at provider and consumer side. Our approach in contrast does not demand any modifications at the cloud service provider side. The compliance enforcement of our approach is solely performed within the federation service and thus can be seen as a light-weighted transparent layer on top of unmodified cloud storage services.

Approaches like [6], [7] are related to our work as they enforce compliance in a cloud storage context. They differ from our approach as they enforce compliance demands at the storage-provider side, which implies modification of the infrastructure.

VI. Discussion and Conclusion

Despite sufficient performance and security in the anticipated application area, our approach has some limitations. The enforcement is limited to events within the federation service itself. Once data leaves this environment, compliance requirements cannot be enforced anymore. Also, the imprecision of the employed geolocation algorithm highly influences the enforcement and may lead to wrong distribution decisions. Furthermore the specification of compliance policies is not trivial due to the underlying specification formalism and the inherently blurry, non-binary, and potentially conflicting nature of compliance regulations. Our approach may have side effects on important quality aspects of the federation service like data availability. Considering inevitable cost-benefit tradeoffs between compliance and availability, policies must thus be carefully balanced out in this respect.

References

[1] D. Bermbach, M. Klems, M. Menzel, and S. Tai, "Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs," in *Proc. of CLOUD'11*, 2011.

[2] A. Pretschner, E. Lovat, and M. Büchler, "Representation-independent data usage control," in *Proc. of STM'11*, 2011.

[3] T. Wüchner and A. Pretschner, "Data loss prevention based on data-driven usage control," in *ISSRE*, 2012.

[4] I. Papagiannis and P. Pietzuch, "Cloudfilter: practical control of sensitive data propagation to the cloud," in *Proc. of CCSW'12*, 2012.

[5] P. Massonet, S. Naqvi, C. Ponsard, J. Latanicki, B. Rochwerger, and M. Villari, "A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures," in *Proc. of IPDPSW'11*, 2011.

[6] S. Betge-Brezetz, G.-B. Kamga, M. Ghorbel, and M.-P. Dupont, "Privacy control in the cloud based on multilevel policy enforcement," in *Proc. of CLOUDNET'12*, 2012.

[7] W. Itani, A. Kayssi, and A. Chehab, "Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures," in *Proc. of DASC'09*, 2009.